State-Space Autoregressive Decoding for Neural Combinatorial Optimization

Ritwika Kancharla [1]

Abstract

Neural combinatorial optimization methods typically rely on attention-based encoder–decoder architectures to construct solutions for routing problems. In this position paper, we argue that such formulations obscure the inherently stateful nature of combinatorial optimization. We propose *state-space autoregressive decoding* as an alternative architectural primitive, explicitly separating static problem structure, dynamic feasibility state, and sequential decision-making. This perspective yields linear-time decoding, persistent constraint tracking, and a closer alignment between neural architectures and classical solver logic. Our goal is to articulate a principled architectural direction for scalable neural optimization rather than to establish state-of-the-art performance.

1. Introduction

Neural combinatorial optimization (Neural CO) has emerged as a promising approach for solving routing and planning problems such as the Traveling Salesperson Problem (TSP) and Vehicle Routing Problem (VRP) (Bello et al., 2016; Kool et al., 2019; Wu et al., 2024). Rather than performing explicit combinatorial search, Neural CO methods aim to *amortize* optimization by learning policies that directly construct solutions from problem instances. This paradigm enables fast inference and has driven rapid progress on canonical routing benchmarks.

Most modern Neural CO methods rely on attention-based encoder–decoder architectures. In these models, a graph or sequence encoder embeds the problem instance, while an autoregressive decoder selects solution elements by attending over node embeddings and partial solutions (Kool et al., 2019; Kwon et al., 2020; Li et al., 2021). Attention has proven effective for modeling pairwise interactions and has thus become the dominant architectural choice in the field. However, this dominance has also shaped how routing problems are framed—primarily as sequence-to-sequence prediction tasks rather than as stateful optimization processes.

This framing introduces a structural mismatch. Routing problems are inherently *stateful*: feasibility constraints such as node visitation, vehicle capacity, and time windows evolve deterministically as decisions are made. Classical solvers explicitly maintain solver state to track these constraints. In contrast, attention-based neural decoders reconstruct partial solution information implicitly at each decoding step through global attention, entangling memory, reasoning, and feasibility handling (Wu et al., 2024; Angioni et al., 2025). As problem size and decision horizon grow, this implicit treatment of state becomes increasingly inefficient and difficult to scale.

In parallel, structured state-space models (SSMs) have emerged as an alternative to attention for long-sequence modeling (Gu et al., 2022; Gu & Dao, 2023). Rather than relying on token-to-token interactions, SSMs maintain a compact latent state that evolves through linear-time updates. This design enables constant-memory inference and stable long-horizon reasoning, making SSMs well suited for problems that require persistent memory and predictable computational cost.

In this paper, we argue that routing and related combinatorial optimization problems are more naturally modeled as *stateful dynamical systems* than as pure sequence modeling tasks. We propose *state-space autoregressive decoding* as an architectural primitive for Neural CO. The core idea is to explicitly separate three components that are conflated in attention-based decoders: (i) static problem structure, (ii) dynamic feasibility state, and (iii) sequential decision-making. Static structure is encoded once per instance, while a persistent latent state evolves across decoding steps to track constraint-relevant information. Decisions are produced by querying this evolving state against the static instance representation.

The goal of this work is not to establish new state-of-the-art performance or to replace highly optimized classical solvers. Instead, we aim to articulate a principled architectural direction for scalable, constraint-aware neural optimization. By aligning neural architectures more closely with the logic of classical solvers—while retaining the benefits of amortized inference—we hope to provide a foundation for future work

[1]Independent Researcher. Correspondence to: Ritwika Kancharla <ritwikareddykancharla@gmail.com>.

Preprint. December 18, 2025.

on large-scale, dynamic, and online routing problems.

2. Neural Combinatorial Optimization at Scale

2.1. Amortized Optimization

Neural combinatorial optimization (Neural CO) approaches routing and planning problems by learning *amortized solvers* over a distribution of problem instances (Bello et al., 2016; Kool et al., 2019). Rather than solving each instance from scratch via explicit combinatorial search, a neural model is trained to directly construct high-quality solutions through a fixed sequence of learned operations. At inference time, optimization is performed implicitly by executing the learned policy, resulting in predictable runtime and fast solution generation.

This paradigm has enabled significant progress on canonical problems such as TSP and CVRP, particularly in settings where many similar instances must be solved repeatedly. By shifting the cost of search to training, amortized optimization offers a favorable trade-off between solution quality and inference efficiency. However, this shift also places greater responsibility on the architecture to internalize problem structure, feasibility constraints, and long-horizon dependencies without relying on iterative refinement or backtracking.

2.2. Attention-Centric Decoding

Most modern Neural CO methods adopt attention-based encoder–decoder architectures (Kool et al., 2019; Kwon et al., 2020; Li et al., 2021). In these models, a graph encoder embeds the problem instance, while an autoregressive decoder uses attention to select solution elements one at a time. At each decoding step, attention is recomputed over the full set of node embeddings, often conditioned on the partial solution constructed so far.

Attention provides a flexible mechanism for modeling pairwise interactions and has proven effective for capturing geometric and relational structure. However, this flexibility comes at a computational cost. Self-attention incurs quadratic time and memory complexity in the number of nodes, and the decoder must repeatedly reconstruct partial solution information at every step. As problem size grows, this repeated global interaction becomes increasingly expensive and limits scalability in both training and inference.

2.3. Architectural Mismatch

Routing problems are inherently *stateful*. Feasibility constraints such as node visitation, remaining vehicle capacity, and time windows evolve deterministically with each decision. Classical solvers explicitly maintain solver state to track these evolving constraints throughout optimization.

In attention-centric Neural CO models, constraint information and partial solution history are represented implicitly through attention weights and distributed activations. Rather than maintaining a persistent state, the model reconstructs relevant information at each decoding step via global attention and masking (Wu et al., 2024; Angioni et al., 2025). This design conflates memory, reasoning, and feasibility handling into a single mechanism.

As the decision horizon increases, this implicit treatment of state becomes problematic. Long-horizon dependencies must be re-inferred at every step, while feasibility constraints are enforced externally rather than being encoded directly into the model's dynamics. These limitations motivate architectures that treat routing as a stateful optimization process, with explicit mechanisms for persistent memory and constraint evolution.

3. Routing as a Stateful Optimization Process

3.1. Problem Setting

We consider routing problems such as the Traveling Salesperson Problem (TSP) under a sequential construction perspective. Given a set of N nodes with coordinates $\{v_i\}_{i=1}^{N}$, the objective is to construct a feasible route that visits each node exactly once and minimizes the total travel cost. While the objective function is fixed for a given instance, feasibility constraints evolve deterministically as decisions are made.

Neural combinatorial optimization methods typically construct solutions autoregressively. At each step t, the model selects the next node to visit based on the partial route constructed so far. This induces a sequential decision process in which the feasible action space depends on the entire decision history. Nodes that have already been visited become infeasible, and in more general routing settings, additional constraints such as remaining vehicle capacity or time windows further restrict future choices.

From this perspective, routing is not a static prediction problem but a *stateful optimization process*. The state at step t consists of all information required to determine feasibility and evaluate future decisions, including which nodes have been visited and how constraints have evolved. Any model that constructs routes sequentially must therefore represent and update this state, either explicitly or implicitly, as part of its decision procedure (Wu et al., 2024).

3.2. Solver State Perspective

Classical routing solvers explicitly maintain solver state throughout the optimization process. In exact methods and heuristic solvers alike, the solver tracks which nodes have been visited, which constraints are active, and how partial

solutions evolve as new decisions are made. This explicit state enables consistent feasibility enforcement and structured reasoning over long decision horizons.

In contrast, most neural combinatorial optimization models do not maintain an explicit notion of solver state. Instead, partial solution information is encoded implicitly through distributed activations and attention weights within the decoder (Kool et al., 2019; Kwon et al., 2020). Feasibility is typically enforced through external masking rules, while the model is expected to reconstruct relevant historical information at each decoding step.

This implicit treatment of state introduces inefficiencies as problem size and horizon increase. Long-horizon dependencies must be repeatedly inferred rather than accumulated, and constraint-relevant information is not preserved in a persistent representation. These limitations motivate architectures that maintain an explicit latent state that evolves alongside the solution, serving as a differentiable analogue of classical solver state. By modeling routing as a stateful optimization process, neural architectures can more naturally align with the structure of combinatorial solvers while retaining the benefits of amortized inference.

4. Structure–State–Decision Decomposition

We decompose routing problems into three interacting components: *static structure*, *dynamic feasibility state*, and *sequential decision-making*. This decomposition mirrors the organization of classical solvers while providing a clean interface for neural amortization. By explicitly separating these components, we isolate instance-level information from solution dynamics and constraint evolution.

4.1. Static Structure

The *static structure* of a routing problem consists of all information that is fixed for a given instance and does not change during solution construction. For the Traveling Salesperson Problem (TSP), this includes node coordinates and pairwise travel costs. For more general routing problems, static structure may also include customer demands, depot locations, and vehicle capacity limits.

Formally, let $\mathcal{I}$ denote a routing instance with node features $\{v_i\}_{i=1}^{N}$. The static structure is encoded once per instance into a set of node embeddings

$$E = \{e_i\}_{i=1}^{N}, \quad e_i \in \mathbb{R}^D,$$

using a graph-based encoder. These embeddings capture geometric and cost-related information and remain fixed throughout decoding. Crucially, they are independent of the partial solution and do not depend on the sequence of decisions made by the model.

This separation ensures that instance-level information is represented explicitly and efficiently, avoiding repeated recomputation as the solution is constructed.

4.2. Dynamic Feasibility State

In contrast to static structure, routing constraints are inherently *dynamic*. Feasibility at a given step depends on the entire history of decisions. For example, once a node is visited it becomes infeasible, and in constrained routing problems the remaining vehicle capacity or time budget evolves with each action.

We model feasibility through a latent state that evolves across decoding steps. Let x_t denote the decision made at step t, and let h_t denote the model's latent state. The state update can be written abstractly as

$$h_{t+1} = f_\theta(h_t, e_{x_t}),$$

where e_{x_t} is the embedding of the selected node. The latent state acts as a differentiable memory that summarizes partial solution structure and constraint-relevant information accumulated so far.

This formulation is analogous to solver state in classical optimization methods. Rather than reconstructing feasibility information at each step, the model maintains a persistent internal representation that evolves deterministically with decisions. Hard feasibility constraints are enforced externally via masking, while the latent state learns to encode soft structural information such as route progress and ordering.

4.3. Sequential Decision Rule

Given the static structure embeddings E and the current latent state h_t, the model selects the next decision using a simple, explicit decision rule. The latent state is projected to a query vector

$$q_t = g_\theta(h_t),$$

which is used to score candidate nodes via a dot product:

$$\ell_t(i) = q_t^\top e_i.$$

Infeasible actions are removed by masking before selection. The next node is then chosen greedily or via sampling from the masked distribution. This decision rule separates *state evolution* from *action selection*, making the optimization process transparent and interpretable.

Importantly, optimization proceeds without explicit search, branching, or backtracking. The model learns transition dynamics that map states to decisions directly, realizing an amortized optimization procedure with predictable runtime and memory usage.

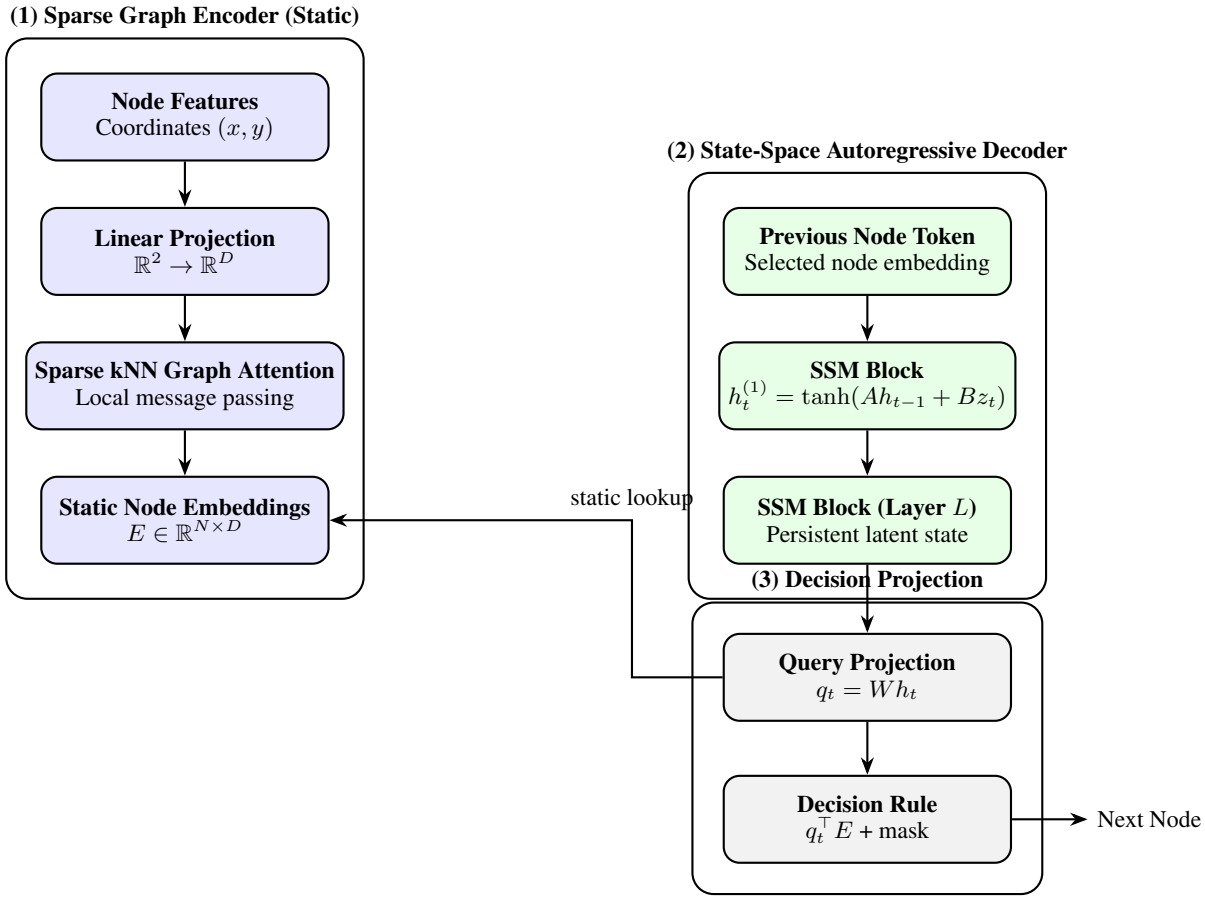

Figure 1. **State-space autoregressive architecture.** A sparse graph encoder produces static node embeddings that represent instance structure. A state-space decoder maintains a persistent latent state that evolves autoregressively with each decision. Next-node selection is performed via an explicit dot-product between the decoder query and static node embeddings, with hard feasibility enforced by masking.

5. State-Space Autoregressive Architecture

The proposed architecture directly reflects the implementation used in our experiments and separates routing into static structure encoding, stateful solution evolution, and explicit decision-making.

5.1. Sparse Graph Encoder

Static instance structure is encoded using a sparse graph neural network. Given node coordinates, each node is first mapped to a D-dimensional embedding via a linear projection. These embeddings are then refined using sparse k-nearest-neighbor attention, which performs localized message passing over a fixed proximity graph.

The encoder is applied once per instance and produces a set of static node embeddings

$$E = \{e_i\}_{i=1}^{N}.$$

These embeddings remain fixed throughout decoding and do not depend on the partial solution or decision history.

5.2. State-Space Autoregressive Decoder

Solution construction is performed by a token-level autoregressive decoder based on a lightweight state-space model. At each decoding step, the decoder receives the embedding of the previously selected node and updates a persistent latent state that summarizes the partial route.

Each decoder layer maintains a latent state h_t that evolves according to

$$h_{t+1} = \tanh(Ah_t + Bz_t),$$

where z_t denotes the input token embedding and A, B are learned diagonal parameters. This update corresponds to a simplified Mamba-style state-space model and enables linear-time decoding without attention over the solution history.

5.3. Decision Projection

At each step, the decoder projects its final latent state to a query vector q_t. Candidate nodes are scored via a dot

product with the static node embeddings,

$$\ell_t(i) = q_t^\top e_i,$$

followed by masking of infeasible actions. The next node is selected greedily at inference time or by sampling during training.

6. Relation to Classical Solvers

Classical approaches to routing problems such as the Traveling Salesperson Problem (TSP) and Vehicle Routing Problem (VRP) rely on exact or heuristic optimization algorithms, including mixed-integer linear programming (MILP), branch-and-bound, cutting-plane methods, and large neighborhood search. These solvers provide strong solution quality and, in some cases, optimality guarantees, but typically require iterative search procedures and substantial computational overhead.

6.1. MILP Intuition

Routing problems admit compact formulations as mixed-integer linear programs. Binary decision variables indicate whether an edge is selected, while linear constraints enforce feasibility conditions such as degree constraints, capacity limits, and subtour elimination. Solving these formulations requires global reasoning over combinatorial constraints and typically involves repeated re-optimization as constraints are added or relaxed (Toth & Vigo, 2014).

From a high-level perspective, MILP solvers maintain an explicit solver state consisting of partial solutions, feasibility status, bounds, and auxiliary constraints. Optimization proceeds through iterative refinement, branching, and constraint generation until convergence.

6.2. Solver State vs. Latent State

A key distinction between classical solvers and neural routing models lies in how solution history and feasibility are represented. Classical solvers maintain explicit solver state, updating constraint satisfaction and solution structure deterministically at each iteration.

In contrast, neural combinatorial optimization models typically do not maintain an explicit notion of solver state. Instead, feasibility and partial solution information are either reconstructed implicitly at each decoding step (e.g., via attention) or enforced externally through masking.

The proposed approach introduces a persistent latent state that evolves autoregressively with each decision. This latent state serves as a differentiable approximation to solver state, encoding information about visitation history and route progression. Unlike explicit MILP state, this representation is learned rather than prescribed and does not guarantee exact constraint satisfaction beyond hard masking.

6.3. Amortized vs. Iterative Optimization

Classical solvers perform *iterative optimization* on a per-instance basis. Each new problem instance requires running a solver from scratch, with runtime determined by the complexity of the instance and the solver configuration.

Neural combinatorial optimization instead adopts an *amortized* perspective. A neural model is trained over a distribution of problem instances to learn a mapping from instance structure to high-quality feasible solutions (Bello et al., 2016; Kool et al., 2019). At inference time, optimization is performed implicitly by executing a fixed sequence of learned state transitions, without branching, backtracking, or re-optimization.

The proposed state-space decoder follows this amortized paradigm. Inference consists of a single autoregressive rollout with predictable runtime and memory usage, independent of solver-specific assumed budgets.

6.4. Scope and Limitations

The proposed approach should be viewed as a learned approximation to routing optimization rather than as a replacement for classical solvers. It does not provide optimality guarantees, nor does it attempt to replicate the full logic of MILP solvers, such as cutting-plane generation or branch-and-bound search.

Instead, the model approximates the behavior of a routing heuristic through a learned, stateful decision process. Its primary advantages lie in inference efficiency, scalability, and suitability for repeated or low-latency deployment. Classical solvers remain preferable in settings where optimality guarantees or instance-specific search are required.

7. Preliminary Evidence

This section presents preliminary experimental results intended to illustrate the qualitative behavior and scaling properties of the proposed architecture. The goal is not to establish state-of-the-art performance, but to demonstrate that state-space autoregressive decoding provides a viable and stable learning signal for routing problems of increasing size.

7.1. Experimental Setting

We evaluate the proposed model on the Euclidean Traveling Salesperson Problem (TSP) with problem sizes $N \in \{10, 20, 50, 100\}$. Node coordinates are sampled independently from the unit square. Models are trained using a self-critical reinforcement learning objective, with greedy

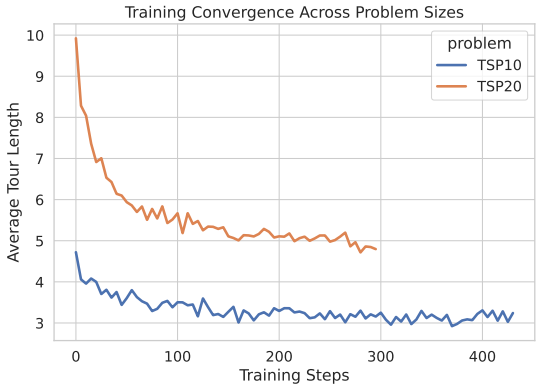

Figure 2. Training convergence.

decoding used for evaluation.

All results are obtained using single-pass autoregressive decoding without beam search, multi-start strategies, or post-processing. These experiments are designed to highlight architectural behavior rather than absolute solution quality.

7.2. Illustrative Results

Figure 2 shows training convergence for increasing problem sizes. Across all values of N, the average greedy tour length decreases steadily with training, indicating that the model learns meaningful routing policies even at long horizons. While convergence becomes slower as N increases, no instability or collapse is observed.

8. Discussion

What this perspective enables. Framing routing as a stateful optimization process enables architectural choices that are difficult to realize within attention-centric neural solvers. By maintaining a persistent latent state that evolves autoregressively, the proposed approach avoids repeated global recomputation over the solution history and supports decoding with predictable time and memory complexity. This property becomes increasingly important as problem size and decision horizon grow.

Moreover, explicit state evolution provides a natural mechanism for internalizing constraint-related information. Rather than reconstructing feasibility at each step via attention or external logic, constraint awareness is accumulated implicitly through state updates. This aligns the model's internal dynamics more closely with classical solver state, while preserving the efficiency benefits of amortized inference.

Limitations. The proposed approach does not provide optimality guarantees and does not perform explicit search, branching, or backtracking. As a result, solution quality may lag behind highly optimized classical solvers or neural methods that rely on multi-start decoding and local improvement heuristics. Constraint satisfaction beyond hard masking is learned rather than explicitly enforced, which may limit robustness in tightly constrained instances.

The experimental evaluation presented in this work is preliminary and is intended to illustrate qualitative behavior rather than establish competitive benchmarks. More extensive empirical comparisons and ablation studies are left for future work.

Why this direction matters. Despite these limitations, state-space autoregressive decoding offers a promising direction for scalable neural optimization. Many real-world routing and planning problems operate under strict latency and memory constraints, require repeated solution generation, or involve long decision horizons where iterative search becomes impractical.

By separating static structure, dynamic feasibility state, and decision-making, the proposed architecture provides a clean foundation for large-scale and online routing settings. We view this work as a step toward neural optimization models that more closely mirror the structure of classical solvers while retaining the flexibility and efficiency of learned amortized approaches.

9. Conclusion

This paper argues that routing problems are more naturally viewed as stateful optimization processes rather than as sequence-to-sequence prediction tasks. Motivated by this perspective, we introduced a state-space autoregressive architecture that separates static instance structure from dynamic feasibility state and sequential decision-making.

The proposed model replaces attention-centric decoding with persistent latent state evolution, enabling linear-time decoding and predictable inference cost. Preliminary experiments on the Traveling Salesperson Problem illustrate that this architecture provides a stable learning signal across increasing problem sizes and supports long-horizon solution construction without explicit search.

We view this work as an architectural contribution rather than a competitive solver. Our goal is to encourage the Neural Combinatorial Optimization community to explore stateful, solver-inspired model designs that better align with the structure of large-scale routing and planning problems. State-space models offer a promising foundation for this direction, particularly in settings where scalability, latency, and persistent solution state are critical.

References

Angioni, M. et al. Constraint handling in neural combinatorial optimization. *arXiv preprint*, 2025.

Bello, I., Pham, H., Le, Q., Norouzi, M., and Bengio, S. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces. *International Conference on Learning Representations*, 2022.

Kool, W., van Hoof, H., and Welling, M. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

Kwon, Y.-D., Choo, J., Kim, B., Yoon, M., Gwon, Y., and Min, S. Pomo: Policy optimization with multiple optima for reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

Li, J., Chen, Z., Wang, J., and Yan, J. Mdam: A multi-decoder attention model for solving routing problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Toth, P. and Vigo, D. *Vehicle Routing: Problems, Methods, and Applications*. SIAM, 2014.

Wu, Y. et al. Neural combinatorial optimization: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.